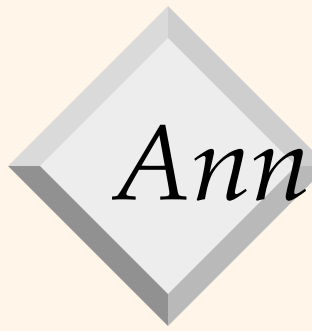
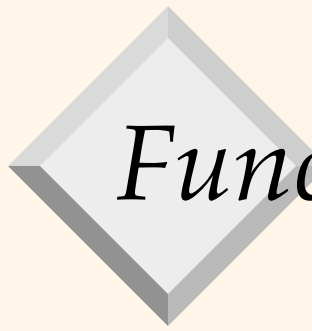


Decompositions & Normal Forms



Announcements/reminders

- ☐ Prelim being graded Friday night
- ☐ H3 is out
 - ER diagrams & Functional Dependencies
 - Material up to and including today
 - Due Friday next week



Functional Dependencies (FDs)

❑ Suppose X and Y are sets of attributes,

❑ Functional dependency $X \rightarrow Y$ holds over R if:

$$\forall t \in R, s \in R, \pi_X(t) = \pi_X(s) \Rightarrow \pi_Y(t) = \pi_Y(s)$$

- given two tuples in R , if their X values agree, then the Y values must also agree

❑ A generalization of keys

Closures

- ☐ Starting set of FDs F
- ☐ The **closure** F^+ of F is the set of all FDs implied by FDs in F

$$F^+ = \{D \mid F \models D\}$$

- ☐ Can be computed using Armstrong's axioms
- ☐ Can be very large



Attribute closures

☐ Typically, we just want to check if a specific FD is in the closure of a set of FDs F .

☐ E.g. suppose

$$F = \{A \rightarrow D, AB \rightarrow E, BI \rightarrow E, CD \rightarrow I, E \rightarrow C\}$$

☐ And I want to know: does F imply $AE \rightarrow D$?



Attribute closures

☐ Want to see whether

$$F \models AE \rightarrow D$$

☐ No need to compute entire closure F^+

☐ Will instead compute the **attribute closure**

☐ Denote this as $(AE)^+$

☐ This is the set of all attributes K such that

$$(AE \rightarrow K) \in F^+$$

☐ Then can just check if $D \in (AE)^+$



Finding all keys of a relation

- ❑ How to do this?
- ❑ Iterate over all subsets X of attributes
 - Check if U (set of all attributes) is in attribute closure of X
 - ❑ If yes, X is a key
 - ❑ If not, X is not a key

Covers

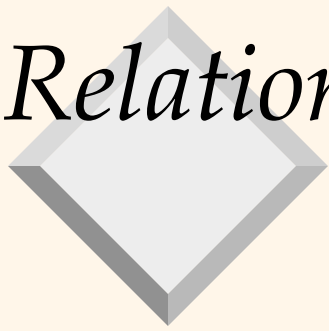
- ☐ Sometimes two different sets of FDs may have the same closure
- ☐ If $F^+ = G^+$ then F is a *cover* for G and vice versa



Summary so far

- ❑ FDs are a marker of redundancy
- ❑ To detect redundancy, need to find all FDs that hold over a relation
- ❑ Armstrong's axioms can help with that
- ❑ We can take a set of FDs and compute:
 - Closures
 - Attribute closures
 - Covers

Relation Decomposition




S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

S	N	L	R	H
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40


Wages

R	W
8	10
5	7



Decomposition

- ☐ A decomposition of R consists of replacing R by two or more relations such that:
- Each new relation scheme contains a subset of the attributes of R (and no attributes that do not appear in R), and
 - Every attribute of R appears as an attribute of one of the new relations.



Decomposition

- ❑ Decompositions can lead to trouble!
- ❑ Performance issues
 - Some queries now require a join
- ❑ Also, some more subtle and/or more serious problems

A Lossy Decomposition

A	B	C
1	2	3
4	5	6
7	2	8



A	B
1	2
4	5
7	2

B	C
2	3
5	6
2	8



A	B	C
1	2	3
4	5	6
7	2	8
1	2	8
7	2	3



Lossless Join Decompositions

□ Decomposition into X and Y is lossless-join w.r.t. a set of FDs F if, for every instance R that satisfies F :

$$\pi_X(R) \bowtie \pi_Y(R) = R$$

□ Note: it is always true that

$$\pi_X(R) \bowtie \pi_Y(R) \supseteq R$$



Lossless Join Decompositions

- ❑ It is essential that all decompositions used to deal with redundancy be lossless!



More on Lossless Join

❓ The decomposition of R into X and Y is **lossless-join** wrt F if and only if the closure of F contains:

- $X \cap Y \rightarrow X$, or
- $X \cap Y \rightarrow Y$

A	B	C
1	2	3
4	5	6
7	2	8

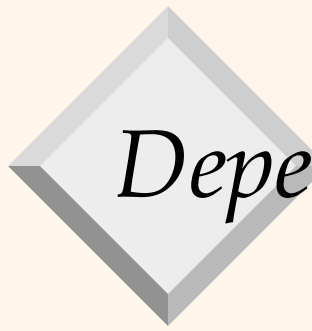


A	B
1	2
4	5
7	2

B	C
2	3
5	6
2	8

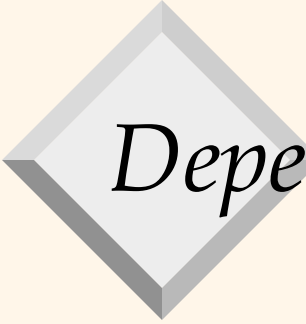
A	B	C
1	2	3
4	5	6
7	2	8
1	2	8
7	2	3





Dependency Preserving Decomposition

- ❑ Consider CSJDPQV, C is key, $JP \rightarrow C$ and $SD \rightarrow P$.
- Decomposition: CSJDQV and SDP
 - Lossless join because SD is key for one of the tables
 - Problem: Checking $JP \rightarrow C$ requires a join!

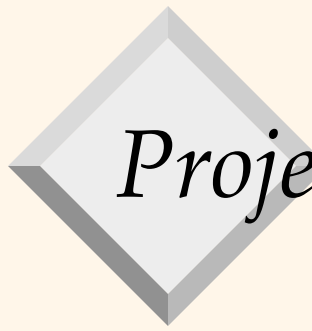


Dependency Preserving Decomposition

[?] Dependency preserving decomposition

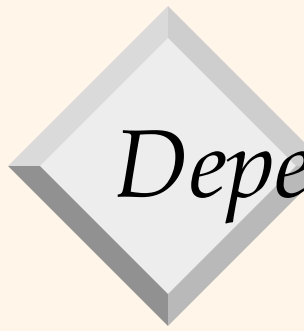
(Intuitive):

- If R is decomposed into X and Y ,
- and we enforce the FDs that hold just on X , and just on Y ,
- then all FDs that were given to hold on R must also hold.



Projection of a set of FDs

- If F is a set of FDs and X a subset of attributes, then the projection of F onto X (denoted F_X) is the set of FDs $U \rightarrow V$ in F^+ such that U, V are in X .
- Intuitively, these are the FDs implied by F that can be enforced by looking at the attributes in X in isolation.



Dependency Preserving Decomposition

☐ A decomposition of R into X and Y is dependency preserving if

$$(F_X \cup F_Y)^+ = F^+$$

☐ That is, if we consider only


- dependencies in the closure F^+ that can be checked in X without considering Y,
- and those that can be checked in Y without considering X,

☐ Then these dependencies imply all FDs in F^+ .



Dependency Preserving Decompositions (Contd.)

- ❑ Dependency preserving does not imply lossless join:
 - ABC, $A \rightarrow B$, decomposed into AB and BC.
 - Not lossless join as shown previously
- ❑ Also, lossless join does not imply dependency-preserving



Decompositions

- ❑ Decompositions remove redundancy
- ❑ But bring problems of their own
 - Some queries require a join
 - May lose information if not careful
 - Checking FDs may require a join



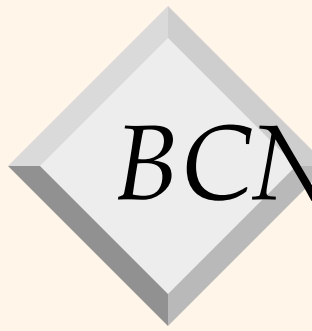
Now finally back to our goal!

- ☐ Want to remove redundancy
- ☐ Given a set of dependencies F , we can compute its closure using Armstrong's axioms
 - So we know all the dependencies that hold
- ☐ We are ready to decompose, hopefully avoiding the major pitfalls of decomposition



Normal Forms

- ❑ The "end goal" of decomposition
- ❑ There is a variety of *normal forms* for relations
 - All have formal definitions
- ❑ Decompose until reach desired normal form
- ❑ Main normal forms of practical interest: BCNF and 3NF



BCNF (Boyce-Codd NF)

- ❑ Motivation: no redundancy due to FDs
- ❑ Only FDs allowed are keys

BCNF

- ☐ A relation R with FDs F is in BCNF if, for all $X \rightarrow A$ in F^+ (X = set of attrs, A = single attr.) either:
- $A \in X$ (called a *trivial* FD), or
 - X contains a key for R.
- ☐ The only non-trivial FDs that hold over R are key constraints.



Decomposition into BCNF

☐ If $X \rightarrow A$ is in closure of F and violates BCNF, decompose R into $R - A$ and XA , repeating if needed



Example decomposition

- ❑ CSJDPQV, key C, $JP \twoheadrightarrow C$, $SD \rightarrow P$, $J \rightarrow S$
- ❑ To deal with $SD \rightarrow P$, decompose into SDP, CSJDQV.
- ❑ To deal with $J \rightarrow S$, decompose CSJDQV into JS and CJDQV
- ❑ Order in which we process FDs may lead to different decompositions



Decomposition into BCNF

- ❑ Consider relation R with FDs F . If $X \rightarrow A$ is in the closure of F and violates BCNF, decompose R into $R - A$ and XA , repeating if needed
- ❑ Is this guaranteed to terminate?
- ❑ Is this guaranteed to produce a lossless-join decomposition?

Reminder

☐ The decomposition of R into X and Y is **lossless-join** wrt F if and only if the closure of F contains:

- $X \cap Y \rightarrow X$, or
- $X \cap Y \rightarrow Y$

☐ In particular, the decomposition of R into UV and R - V is lossless-join if $U \rightarrow V$ holds over R.

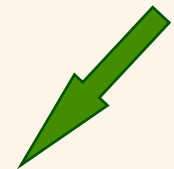
A	B	C
1	2	3
4	5	6
7	2	8

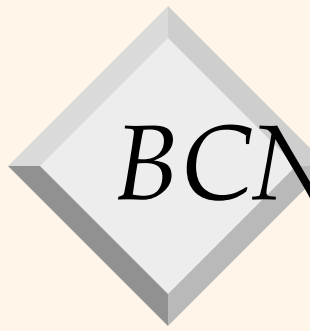


A	B
1	2
4	5
7	2

B	C
2	3
5	6
2	8

A	B	C
1	2	3
4	5	6
7	2	8
1	2	8
7	2	3





BCNF and Dependency Preservation

❓ In general, there may not be a dependency preserving decomposition into BCNF.

- e.g., $SBD, SB \rightarrow D, D \rightarrow B$
- Not in BCNF because D is not a key.
- Can't decompose while preserving $SB \rightarrow D$



Third Normal Form (3NF)

- ☐ Allows slightly more redundancy than BCNF
 - If a schema is in BCNF, it is in 3NF, but not necessarily vice versa
- ☐ There is **always** a dependency-preserving decomposition into 3NF



Third Normal Form (3NF)

- ☐ Reln R with FDs F is in **3NF** if, for all $X \rightarrow A$ in F^+
 - $A \in X$ (called a *trivial* FD), or
 - X contains a key for R, or
 - A is part of some **key** for R.
- ☐ *Minimality* of a key is crucial in third condition above!
- ☐ If R is in BCNF, obviously in 3NF.



The gap between 3NF and BCNF

- ❑ Reserves SBDC, $S \rightarrow C$, $C \rightarrow S$, key SBD.
- ❑ This is in 3NF (because CBD is also a key)
- ❑ But for each reservation of sailor S, same (S, C) pair is stored.



Decomposition into 3NF

- ❑ Obviously, the algorithm for lossless join decomp into BCNF can be used to obtain a lossless join decomp into 3NF (typically, can stop earlier).
- ❑ To ensure dependency preservation, one idea:
 - If $X \rightarrow A$ is not preserved, add relation XA .
 - Problem is that XA may violate 3NF!
- ❑ Refinement: Instead of the given set of FDs F , start with a *minimal cover for F* .



Minimal Cover for a Set of FDs

❓ Minimal cover G for a set of FDs F :

- Closure of F = closure of G .
- Right hand side of each FD in G is a single attribute.
- If we modify G by deleting an FD or by deleting attributes from an FD in G , the closure changes.

❓ Intuitively, every FD in G is needed, and “*as small as possible*” in order to get the same closure as F .



Minimal Cover Example

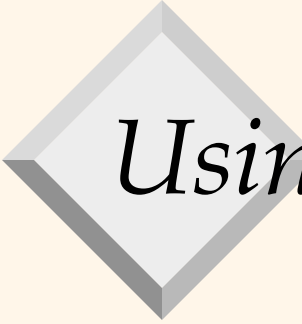
❑ Algorithms to compute minimal cover are known
(see textbook)

❑ For example

$$F = \{A \rightarrow B, ABCD \rightarrow E, EF \rightarrow G, EF \rightarrow H, ACDF \rightarrow EG\}$$


❑ This has a minimal cover

$$G = \{A \rightarrow B, ACD \rightarrow E, EF \rightarrow G, EF \rightarrow H\}$$



Using minimal covers for 3NF

- ❑ Compute G to be the minimal cover of the original set F
- ❑ Compute a lossless join decomposition using algorithm similar to BCNF
- ❑ If some $X \rightarrow A$ in G is not preserved, add XA to schema
- ❑ This is guaranteed to satisfy 3NF
- ❑ Explanation why in textbook if you're interested.



Summary of Schema Refinement

- ❑ BCNF implies free of redundancies due to FDs
- ❑ If a relation is not in BCNF, we can try to decompose it into a collection of BCNF relations.
- ❑ If a lossless-join, dependency preserving decomposition into BCNF is not possible, consider 3NF
- ❑ Decompositions should be carried out and/or re-examined keeping *performance issues* in mind